Streaming Query Driven Architecture for general Wireless Sensor Networks

Prakhar Banga Vineet Hingorani
Supervisor: Prof. Sumit Ganguly

February 7, 2013

Dept. of Computer Science and Engineering,
IIT Kanpur

Abstract

1 Introduction

Wireless Sensor Networks have a wide range of applications in both the static(from area monitoring to agriculture-based) and dynamic(from military based applications to animal tracking) scenarios. The primary difference between a WSN and a normal wired network is the resource constraints in WSNs. The constraints here are in terms of Power consumption, Computational resources, Memory usage, etc. The sensor nodes in network are very short on their power resource and henceforth their lifetime decreases if they are always sending data. Most of the information in sensor nets is not needed until something ill has happened, in the sense WSNs are event-driven. For example, in a plant health monitoring system, the data sensed by the sensor node(which is normal most of the time) is not needed till the level of plant health has gone below a threshold level(which is a rare event). This problem can be handled by providing the nodes with some computation power and the nodes send their data only when an anomaly has occurred. Presently, such WSNs have been deployed at huge levels but all of them are domain specific. Development of a WSN framework is usually application dependent. Also if the program running on a particular node has to be changed, it has to be changed manually. For a real simulation, we need to have a prototype network of WSNs to test the design of our framework.

2 Problem statement

We would like to design a general framework which facilitates and automates a lot of the manual work done in the deployment and maintenance of WSNs. We want to have a framework such that it is general enough to be used for a lot of applications of WSNs. It should also be programmable so that the job the WSN performs can be changed anytime. Moreover, it should be remotely programmable, so that it can be programmed from a central location. It would also be nice to have the programming paradigm as easy and simple as possible for the programmer. Currently most of these problems are handled manually and in an ad-hoc manner. A construction of such a general framework would be analogous to the systematization of databases which happened in the 1960s and 70s, when

3 Applications

WSNs have a wide range of applications. We classify them in two broad areas:

- Static WSNs: All nodes have a fixed location. e.g.
  - Environment monitoring: Environment monitoring usually involves measuring critical aspects of the environment, such as the temperature, humidity, air pressure, radiant light etc. It has a wide impact on the understanding of the environment and habitat dynamics.
  - Smart buildings: A ‘smart building’ is a building which can automate tasks which humans perform. Human presence sensors and motion sensors can be used for the common job of automatically switching the lights and the air conditioner on and off depending on human presence. Less common tasks like detecting excreta can be used in buildings with animals.
  - Mechanical monitoring: Mechanical monitoring is the monitoring of physical structures, like machines and bridges. Vibrational patterns of these structures can indicate wear-and-tear and can be used to detect upcoming failure.
- Dynamic WSNs: Nodes don’t have a fixed location. e.g.

- Military monitoring: Dynamic WSNs have wide applications in the military. One application includes monitoring soldier locations which can be critical in times of war or when the soldiers are in certain high-risk areas. Another application is vehicle surveillance, in which military vehicles are fitted with surveillance equipment and they move around collecting data.
- Animal monitoring: Scientists in their quest to understand the habitat dynamics of certain animals, use trackers to collect data about their whereabouts throughout the year. Such monitoring is done very frequently for bird and fish species and yields a lot of interesting results about their year-round migration.
- Vehicle tracking: GPS can be used to track vehicles and maintain the locations of a number of vehicles. This can be used by transport companies, in healthcare by tracking ambulances etc. Of course, the possibilities for WSN applications are endless and any application can be augmented to do other tasks as well.

4 Proposal

The proposed architecture is something like this:

1. The whole system is a hierarchical control system(**??**)
2. The system doesnt assume that nodes and sensors have unique ids(IP addresses, MAC addresses etc.)
3. Every node and sensor has a set of properties, which are used to diffuse tasks/queries through the network. Therefore, all multicast is property-based.
4. The tasks/queries are written in an SQL-like language

5 Programming Paradigm

In most of the cases the sensor networks are deployed and then any change in the working of sensors need reprogramming in them. There is a need of on-the-fly programming of sensor nodes in the sensor networks. Manual reprogramming of a network with thousands of nodes is infeasible. A lot of work has been done in the field of programming of these sensors implementing various wireless routing algorithms that would be useful in such a resource constraint scenario.

There are loads of issues in designing a Programming framework for WSNs. Some of them are:

a) Program Length: The packets needed to send a large program code would be high and the network congestion increases if large number of packets are send. Also, the power constraints make such communication difficult for the multi-hop sensors. The compute power and memory is also another constraint which will come into place if the Program Code is very big.

b) Binary Program: Sending binary program also makes the above issues worse. As said before, the computer power of sensor nodes is very less. Having a compiler at each of the nodes individually is not feasible in a large sensor network and therefore we need to send the large binary code.

c) Version Control: The dynamic reprogramming have this issue of different versions for programs at various nodes. This version control can be done at the administrator or at the nodes itself.The design of the framework should be such that the whole network should be scalable and reliable keeping in mind the constraints.

d) Programming Interface: The interface for the administrators is also an important issue. It should be easier for a non-programmer to send queries for particular nodes in the network.

e) Initiation: There are basically two schemes proposed in the literature of programming these sensors[1]:- 1. Code Dissemination- Initiated by the network administrator where code is send to the individual sensors or group of sensors. This could be for various purposes like fixing bugs, changing network structure, changing programs, etc. 2. Code Acquisition- This is initiated by sensor nodes themselves as per need or per changing surrounding. The choice depends on what gives you the best performance in overall ways.

6 Networking

Wireless Routing has been a great subject of study for the past many years. Issues like reliability, security, quick transmission, etc. are the need for the wireless communication protocols. In the case of WSNs, the requirements for these protocols increase more than mere Wireless networking. Some of the issues faced in routing the packets and networking in WSNs are:

- Hops: To make a general framework the network should be able to handle both the single-hop and multihop transmissions. For remote sensor networks applications, multi hop reprogramming is a need. There are power constraints in nodes that are used frequently to transmit the programs from administrator to far flung nodes in multi hop nets.

- Scheduling: Scheduling issues is also a great concern for reliable and quick transmission. To handle the Hidden Terminal problem we cant use the conventional solution due to the resource constraints. Avoiding as many collisions as possible is basically the need.

- Pipelining: The framework should be able to pipeline the packets as much as possible. As the packet payload size is very less for, the time taken to send the program to a node 3 hops away from administrator without pipelining would take factors of time more than what it takes with pipeline.

- Addressing: To direct the code to the selected group of nodes, there should be some mechanism for that. Scope Selection[1][2] is one of the scheme that is done before transmitting the actual program to direct it towards the nodes. Addressing the nodes could also be done. This is an important issue of how to address the nodes and the intricacies behind it.

7 OS Requirements

8 Related Work

A lot of work has been done in network programming mechanisms. These basically address the the major problems of how to program?, how to send? and where to send? We will briefly address some of the past work done in this area.

Motes developed in Berkeley are used extensively for network programming. XNP[3]is an implementation for TinyOS that broadcasts the whole program to the network and is a single-hop network. The sensor nodes accept the code depending on their needs individually. MOAP[4] is a multi hop improvement over the XNP. The code dissemination algorithm used in MOAP sends the packets to a few number of nodes rather than flooding the whole network. Deluge[5], MNP[6] and Aqueduct[7] does pipelining for sending the packets and improves upon the MOAP. While Deluge and MNP are for the whole network, Aqueduct is for selected number of nodes. Reijer's approach[10] doesn't support pipelining but is a platform dependent scheme using Mate's Virtual Machine Code. Instead of sending the Native code, it sends a VM code that is application specific. This improves upon the number of packets sent but has a disadvantage of computation of the binary mapping at the sensor nodes. It works in a single hop network and floods the network. Trickle[8] is an improvement over Reijer's that works on multi-hop networks and doesn't broadcast i.e. flood the network. Incremental[11] approach given by Culler et al. is an approach that updates the programs at the sensor nodes in an incremental manner in such a way that the difference between new program image and old version is computed and writes in the memory are done. This uses a modified version of Rsync algorithm that would work on constraint sensors. TinyCubus[9] is a scheme that instead of sending the whole program sends the modular updates in the program to the sensor nodes. This improves a lot over the past implementations in terms of power consumption and network communication. The only issue here is that generally programs are not much modular and it is difficult to change the programs in such a way.

The frameworks given described till assume that there is no hierarchy in the network. Firecracker[12] works for a hierarchichal network and is a multi-hop framework. All of these frameworks have been designed for Berkeley Motes and work on TinyOS environment with CSMA as MAC Layer protocol. Sprinkler[13] is a framework using TDMA as MAC protocol for multi-hop networks and is not designed for the Motes in particular.

The designed frameworks can be put in different categories as described above in the various sections. We will be aiming towards a general framework with least assumptions for the network.

9 Applications

References:

[1] Reprogramming Wireless Sensor Networks

[2] Incremental Network Programming for Wireless Sensors

[3] Mote In Network Programming User Reference

[4] A Remote Code Update Mechanism for Wireless Sensor Networks

[5] DELUGE

[6] MNP